Adv. Studies Theor. Phys., Vol. 7, 2013, no. 8, 359 - 361
HIKARI Ltd, www.m-hikari.com

A Genetic Algorithm with Expansion Operator for the 3-Satisfiability Problem

Vladimir Popov

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
Vladimir.Popov@usu.ru

Abstract

It is well known that there are many problems that standard genetic algorithms fail to solve. There are a number of refinements of standard genetic algorithms that can be used to solve hard problems. In this paper we consider genetic algorithms with expansion operator for the 3-satisfiability problem.

PACS: 02.70.Rr

Keywords: genetic algorithms, expansion operator, satisfiability

Genetic algorithms are a class of algorithms inspired by the biological process of evolution by natural selection. Usually, genetic algorithms evolve a population of solutions by iteratively applying some search operators. Theoretical evidence suggests that the success of genetic algorithms stems from their ability to combine building blocks from separate individuals in order to form better solutions. However, it is well known that there are many problems that standard genetic algorithms fail to solve. An idea of refinement of standard genetic algorithms was proposed in [1]. The idea is to preserve good building blocks found by the genetic algorithm. For this purpose, we can use some constraints on the choice of recombination. In particular, an expansion operator can be used [1].

Recently, genetic algorithms for the 3-satisfiability problem is extensively studied and used for solution of other problems (see e.g. [2] – [6]). In this paper we consider genetic algorithms with expansion operator for the 3-satisfiability problem.

In [1], an expansion operator was considered for the shortest common superstring problem. In particular, an expansion operator is an addition of another block to increase the individual's genome length by one block. The added block is selected by expansion operator in such a way that the genome will still be a subsequence of at least one solution.

Let $f(z[1],\dots,z[m])$ be a Boolean function. Let $W=\{u[1],u[2],\dots,u[n]\}$, $u[i]\in\{0,1,*\}^+$, $1\le i\le n$, be a population of chromosomes. We assume that $u[i]=u[i,1]\dots u[i,m]$, $u[i,j]\in\{0,1,*\}$, $1\le i\le n$, $1\le j\le m$.

We can consider $u[i]$ as a partial solution for $f(z[1],\dots,z[m])$. In particular, we assume that $z[j]=u[i,j]$ if $u[i,j]\in\{0,1\}$. We consider only 3CNFs. So, we can assume that $f=\wedge_{j=1}^{k}C[j](z[1],\dots,z[m])$ where $C[j](z[1],\dots,z[m])$ is a clause. We say that $u[i,j]$ is a true assignment for $C[k](z[1],\dots,z[m])$ if $u[i,j]\in\{0,1\}$ and $z[j]=u[i,j]$ evaluates $C[k](z[1],\dots,z[m])$ to true. Let $C(u[i,j])$ be the set of clauses such that $C[k](z[1],\dots,z[m])\in C(u[i,j])$ if and only if $u[i,j]$ is a true assignment for $C[k](z[1],\dots,z[m])$. If $u[i,j]=*$, then we assume that $C(u[i,j])=\emptyset$.

At first, we consider a standard genetic algorithm (SGA) for the 3-satisfiability problem. A proportion $\mathcal{P}$ of the existing population is selected to breed a new generation during each successive generation. We assume that $\mathcal{P}=\lceil\frac{n}{2}\rceil$. Individual chromosomes are selected by a fitness function $\mathcal{F}$. To generate a second generation population of chromosomes, we can use two genetic operators: crossover $\mathcal{C}$ and mutation $\mathcal{M}$. Usually, crossover defines a part of parent chromosome which used for construction of child chromosome. As $\mathcal{C}$ we use standard random operator. If $u[i]$ and $u[j]$ are two parent chromosomes, then we obtain two child chromosomes:

$$\mathcal{C}(u[i],u[j])=u[i,1]\dots u[i,\mathcal{C}(u[i])]u[j,\mathcal{C}(u[i])+1]\dots u[j,m],$$

$$\mathcal{C}(u[j],u[i])=u[j,1]\dots u[j,\mathcal{C}(u[j])]u[i,\mathcal{C}(u[j])+1]\dots u[i,m].$$

For $u[i[1]],\dots,u[i[p]]$, we consider

$$\mathcal{C}(u[i[1]],u[i[2]]),\mathcal{C}(u[i[2]],u[i[1]]),\dots,$$

$$\mathcal{C}(u[i[p-1]],u[i[p]]),\mathcal{C}(u[i[p]],u[i[p-1]])$$

if $p=2q$,

$$u[i[1]],\mathcal{C}(u[i[2]],u[i[3]]),\mathcal{C}(u[i[3]],u[i[2]]),\dots,$$

$$\mathcal{C}(u[i[p-1]],u[i[p]]),\mathcal{C}(u[i[p]],u[i[p-1]])$$

if $p = 2q + 1$. This generational process is repeated until a termination condition $\mathcal{T}$ has been reached. As $\mathcal{T}$ we consider time function. We assume that $\mathcal{F}(u[i]) = |\cup_{j=1}^{m} C(u[i,j])|$. Let $\mathcal{M}$ be a random function which with small probability changes values of $u[i,j]$.

Now, we consider a genetic algorithm with an expansion operator (GAE). The only difference with SGA consists in the usage the following expansion operator instead $\mathcal{C}$. Let $\mathcal{E}(u[i],u[j]) = v[1]\dots v[m]$ where

$$v[k] = \begin{cases} u[i,k], |C(u[i,k])| > 0, \\ u[j,k], |C(u[i,k])| = 0, |C(u[j,k])| > 0, \\ u[i,k], |C(u[i,k])| = 0, |C(u[j,k])| = 0. \end{cases}$$

Selected experimental results are given in Table 1.

average number of generations	10^2	10^3	10^4	10^5
average number of true clauses for SGA	53 %	62 %	69 %	73 %
average number of true clauses for GAE	59 %	72 %	81 %	89 %

Table 1: Experimental results for SGA and GAE.

ACKNOWLEDGEMENTS. The work was partially supported by Analytical Departmental Program "Developing the scientific potential of high school" 8.1616.2011.

References

[1] A. Zaritsky and M. Sipper, The preservation of favoured building blocks in the struggle for fitness: The puzzle algorithm, *IEEE Transactions on Evolutionary Computation*, 8 (2004), 443-455.

[2] A. Gorbenko and V. Popov, SAT Solvers for the Problem of Sensor Placement, *Advanced Studies in Theoretical Physics*, 6 (2012), 1235-1238.

[3] A. Gorbenko and V. Popov, Clustering Algorithm in Mobile Ad Hoc Networks, *Advanced Studies in Theoretical Physics*, 6 (2012), 1239-1242.

[4] A. Gorbenko and V. Popov, The Problem of Finding Two Edge-Disjoint Hamiltonian Cycles, *Applied Mathematical Sciences*, 6 (2012), 6563-6566.

[5] A. Gorbenko and V. Popov, Footstep Planning for Humanoid Robots, *Applied Mathematical Sciences*, 6 (2012), 6567-6571.

[6] A. Gorbenko and V. Popov, Task-resource Scheduling Problem, *International Journal of Automation and Computing*, 9 (2012), 429-441.

Received: February 12, 2013